YOUR FINAL TASK

COMPUTER PROGRAMMING 2

(Week 9)

NAME: Aidre Love S. Cabrera GRADE & SECTION: 12 – STEM A TEACHER: Anthony Pleños DATE SUBMITTED: April 2, 2022

A. Instructions: Write **True** on the blank provided if the statement is correct, otherwise, write **False**.

False 1. Classes that help handle errors in Java are called error classes.

"Error clasess does not handle errors. However, exception classes makes it possible through trycatch to handle unchecked and checked type of errors."

True 2. It is possible to have several catch blocks following a try block.

"We can have multiple catch blocks for single try block. But only one try block is allowed"

False 3. A try block does not need to have a matching catch block.

"There are three premutations of try/catch/finally block: try...catch; try...catch...finally; and try...finally. Therefore, it is not necessary that each try block must be followed by a catch block. However, it iss needed that it must be followed by either a catch block or a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method."

False 4. A catch block does not need to have a matching try block.

"It is necessary for a block to be catch paired with a try block."

True 5. Several catch statements following a single try statements should handle different exceptions.

"Every catch statements must handle various exceptions to avoid redundancy. Otherwise, it will return " java: exception java.lang.[type of sub-exception] has already been caught"

False 6. The finally statement is required after using try and catch statements.

"try-catch statement can run without being paired with finally block."

True 7. The block within the finally statement will be executed regardless of whether or not an error is encountered.

"Before the method is complete, finally block will always run after the try and any catch block. Therefore, finally block executes regardless of whether an exception is thrown or caught."

True 8. The IOException class handles errors that occur during input and output.

"IOException is a Java exception that occurs when an IO operation fails."

True 9. The Exception class handle all types of exceptions.

"Since Exception is the base class of all exceptions, it will catch any exception. Exception is like a 'catch all' exception handler that is a broader type of exception handler. While those sub-exception handers such as ArrayIndexOutOfBoundsException is a specific type of a handler. Therefore, any exception that may get thrown is an Exception"

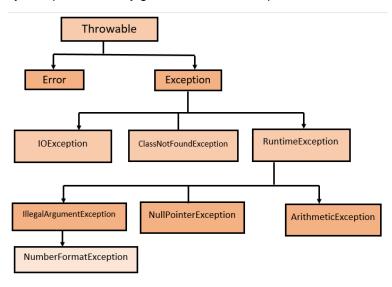


Figure 1. Types of Exceptions

False 10. Declaring catch blocks before a block that handles an Exception class to handle other types of errors would be redundant.

"catch blocks before an Exception class that will handle other unknown exceptions is not redundant. catch exceptions before the Exception class is defined for the purpose of specifity. While Exception class is defined in order to handle other errors that are not handled specifically.

```
public class test {
  public static void main(String[] args) {
    int[] nums = new int[5];
    try {
        System.out.println("Try 1");
        nums[10] = 25;
        System.out.println("Try 2");
    } catch (ArrayIndexOutOfBoundsException ex) {
        System.out.println("Catch 1");
        //Exception that will catch other exceptions
    } catch (Exception ex) {
        System.out.println("Catch 2");
    }
}

4  }

5  }

6  }

7  System.out.println("Catch 1");
    //Exception that will catch other exceptions
    } catch (Exception ex) {
        System.out.println("Catch 2");
    }
}
```

B. Instructions: Insert the missing keyword to execute code, after try catch, regardless of the result.

int[] myNumbers = {1, 2, 3}; System.out.println(myNumbers [10]); } catch (Exception e) { System.out.println("Something went wrong."); } finally { System.out.println("The 'try catch' is finished."); }

Solution

```
Solution.java

1 try {
2  int[] myNumbers = {1, 2, 3};
3  System.out.println(myNumbers[10]);
4 } catch (Exception e) {
5  System.out.println("Something went wrong.");
6 } finally {
7  System.out.println("The 'try 'catch' is finished");
8 }
```

C. Instructions: Give the output produced by the following code snippets.

```
1. int n = 5;
    try {
        n = n / 0;
    }
    catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception Caught");
    }
    catch (NumberFormatException e) {
            System.out.println("Number Format Exception Caught");
    }
    finally {
            System.out.println("Done")
    }
}
```

Answer:

```
java: ';' expected; Line #
```

Realistically, the provided code will result a syntax error. That is because in Line 12, in finally block, the print method has a missing semicolon. Therefore, the output that will be produced when strictly following the provided code snippet will be "java: "expected; Line 11."

Proof of Error

```
Terminal Output

1 // C1 Code Snippet
2 static void sectionC1() {
3    int n = 5;
4    try {
5        n = n / 0;
6    } catch (ArithmeticException e) {
7        System.out.println("Arithmetic Exception Caught");
8    } catch (NumberFormatException e) {
9        System.out.println("Number Format Exception Caught");
10    } finally {
11        System.out.println("Done");
12    }
13 }
14
15 /*
16
17 Terminal Output:
18 Arithmetic Exception Caught
19 Done
20
21 */
```

To alleviate the simple error, we can simply add a semicolon before the error line and the output will be:

Arithmetic Exception Caught Done

C2 Final Answer and Proof

```
Terminal Output

1 // C1 Code Snippet
2 static void sectionC1() {
3    int n = 5;
4    try {
5         n = n / 0;
6    } catch (ArithmeticException e) {
7         System.out.println("Arithmetic Exception Caught");
8    } catch (NumberFormatException e) {
9         System.out.println("Number Format Exception Caught");
10    } finally {
11         System.out.println("Done")
12    }
13 }
14
15 /*
16
17 Terminal Output:
18 G:\Github Projects\src\fileContainers\week9.java:25:43
19 java: ';' expected;Line 11
20
21 */
```

Answer:

```
java: ';' expected; Line #
```

The same error is encountered in this snippet from the previous snippet. However, the solution is the same.

Proof of Error

```
Terminal Output

1 // C2 Code Snippet
2 static void sectionC2() {
3    int n = 5;
4    try {
5         n = n / 0;
6    }
7    catch (Exception e) {
8         System.out.println("Exception Caught");
9    }
10    finally {
11         System.out.println("Done")
12    }
13 }
14
15 /*
16
17 Terminal Output:
18 G:\Github Projects\src\fileContainers\week9.java:25:43
19 java: ';' expected;Line 11
20
21 */
```

C2 Final Answer and Proof

Exception Caught Done

```
Terminal Output

1 // C2 Code Snippet
2 static void sectionC2() {
3    int n = 5;
4    try {
5         n = n / 0;
6    }
7    catch (Exception e) {
8        System.out.println("Exception Caught");
9    }
10    finally {
11        System.out.println("Done");
12    }
13 }
14
15 /*
16
17 Terminal Output:
18 Exception Caught
19 Done
20
21 */
```

Appendix I

```
1 package fileContainers;
 3 public class week9 {
       static void sectionB() {
           try {
                int[] myNumbers = \{1, 2, \overline{3}\};
               System.out.println(myNumbers[10]);
           } catch (Exception e) {
               System.out.println("Something went wrong.");
           } finally {
               System.out.println("The 'try 'catch' is finished");
       static void sectionC1() {
           int n = 5;
           try {
               n = n / 0;
           } catch (ArithmeticException e) {
               System.out.println("Arithmetic Exception Caught");
           } catch (NumberFormatException e) {
                    System.out.println("Number Format Exception Caught");
           } finally {
                   System.out.println("Done");
       static void sectionC2() {
           int n = 5;
               n = n / 0;
           catch (Exception e) {
               System.out.println("Exception Caught");
           finally {
               System.out.println("Done");
       public static void main(String[] args) {
           System.out.println("Section B\n");
           sectionB();
           System.out.println("\nSection C, #1\n");
           sectionC1();
           System.out.println("\nSection C, #2\n");
           sectionC2();
49 }
```

Appendix II

```
1 //Console Output from Intellij
2
3 /*
4
5 Section B
6
7 Something went wrong.
8 The 'try 'catch' is finished
9
10 Section C, #1
11
12 Arithmetic Exception Caught
13 Done
14
15 Section C, #2
16
17 Exception Caught
18 Done
19
20 Process finished with exit code 0
21
22 */
```